Solution 1:

1) NNs can transform the original raw data into a more compact and structured representation that captures relevant information or characteristics (= representation learning). In addition, this compressed representation facilitates the use of "traditional" algorithms for both regression and classification problems.

2) In the case of the first layer (so-called input layer), these neurons simply receive the input (e.g. the input features X) and pass it on to the first of an arbitrary number of so-called hidden layers

3) Their output cannot be accessed inside the network as it is simply passed on to the next layer.

4) Firstly, an affine transformation is performed on the input that is passed on from the previous layer. In many cases, this affine transformation is nothing more than the weighted sum of the input, whereby the weights belong to the parameters that are optimized within the neural network. The thereby transformed input is then further plugged into an activation function. Depending on the choice of function, it introduces a certain degree of non-linearity in order to derive at the new representation of the data.

5) The number of neurons depends on the number of distributional parameters we want to estimate. For example, if we only want to estimate the mean (e.g. in the context of a traditional regression problem), we only need one neuron in the output layer. However, in the context of a multiclass classification problem we need one neuron for each class, as we want the estimated probability for each of these classes. In the context of an regression/classification problem the type of activation function used in the output layer i.a. depends on the assumed data distribution of the target variable y and thereby often corresponds to the link-function used within regression approaches.